DOCUMENT RESUME

ED 277 365 IR 012 473

AUTHOR Sanford, David L.; Roach, J. W.
TITLE Using Rule-Based Computer Programming to Unify Communication Rules Research.
PUB DATE 1 May 86
NOTE 21p.; Paper presented at the Annual Convention of the Eastern Communication Association (Atlantic City, NJ, May 1, 1986).
PUB TYPE Viewpoints (120) -- Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Artificial Intelligence; *Cognitive Processes; *Communication (Thought Transfer); *Communication Research; *Models; Programing Languages; Standards
IDENTIFIERS *PROLOG Programing Language; Theory Development

ABSTRACT

This paper proposes the use of a rule-based computer programming language as a standard for the expression of rules, arguing that the adoption of a standard would enable researchers to communicate about rules in a consistent and significant way. Focusing on the formal equivalence of artificial intelligence (AI) programming to different types of human communication theories, the paper first discusses the natural language processing approach that explicitly tries to model human linguistic and cognitive processes. This discussion examines both AI models and communication-related theories that use allied knowledge structures and those that use contrasting knowledge structures. The paper then introduces PROLOG, the recommended rule-based programming language, and shows the advantages its adoption would have for theory construction and research for those interested in communication rules, and for those interested in developing precise theories from other perspectives. Examples of expressing communication propositions and developing a theory of conversational request structures in PROLOG are given, and approximately 70 references are listed. (DJR)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

ED277365

USING RULE-BASED COMPUTER PROGRAMMING TO UNIFY COMMUNICATION RULES RESEARCH

David L. Sanford
Department of Communication Studies
J. W. Roach
Department of Computer Science

Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

Paper presented to the
Human Information Technologies Interest Group at the
Eastern Communication Association Convention
Atlantic City, NJ, May 1, 1986

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

David L. Sanford

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

IR012473

Abstract

The rules approach to communication is often criticized for failing to generate much empirical research and for having a diversity of approaches. Both criticisms are the result of having no central conception of what rules are, how best to express them, and how to use them to guide research. This paper proposes adopting a rule-based programming language, specifically PROLOG, as the best form for expressing rules of any theory, whether communication rules, cognitivist, or symbolic interactionist theories. Examples of expressing communication propositions and developing a theory of conversational request structures in PROLOG are given. Advantages to adopting PROLOG are discussed.

> In recent years the "rules approach" to studying communication behavior has taken hold. Yet, relatively little in the way of empirical research has been generated. The available research tends to be descriptive in nature and is characterized by a conspicuous absence of prediction along with experimental methods and parametric interpretations of social behavior. (Fairhurst, 1980, p. 1)

This accusation is now a little dated; some empirical work has been done by communication rules researchers (e.g., Pearce, Cronen, Johnson, Jones, & Raymond, 1980; Murdock, Bradac, & Bowers, 1984). Unfortunately, even with these exceptions to Fairhurst's complaint there is no necessary connection between any "rules" and what the research is investigating; the research could have been just as easily generated from a symbolic interactionist or behaviorist perspective.

Another problem often discussed about rules research is the diversity of opinion about what a rule is, how it is known, and what it is for (Delia, 1977; Donohue, Cushman, & Nofsinger, 1980; Shimanoff, 1980). Some argue that rules are like grammar, cognitions controlling actions but not consciously expressible without special training (Frentz & Farrell, 1976), while others argue that they must be explicitly stated (Ganz, 1971) as in Robert's Rules of Order (Robert, 1970). Some say they express a homogeneous social order (Nofsinger, 1974), while others say they express a heterogeneous social order (Cushman & Whiting, 1972). Rules research also differs along other dimensions.

The purpose of this paper is to propose the use of a rule-based computer programming language as a standard for the expression of rules. An obvious advantage to adopting a standard would be to enable researchers to communicate about rules in a consistent and significant way. Other less obvious advantages will be presented below. To explain the advantages of using a rule-based computer programming language, this paper is divided into two major sections. The first section discusses how computer programming, specifically Artificial Intelligence (AI) programming, has a formal equivalence to different types of human communication theories. The second section introduces PROLOG, the recommended rule-based programming language, and shows the advantages its adoption would have on theory construction and research for those interested in communication rules, and for those interested in developing precise theories from other perspectives.

Traditional AI and Communication Theories

AI includes several subtopics, such as problem solving, natural language, theorem-proving, expert systems, and robotics, among others. Natural language processing is a subtopic particularly suited to applications for human communication research. Natural language processing has two important approaches: one approach tries explicitly to model human linguistic and cognitive processes, the other focuses on developing person-machine interfaces with which humans can easily interact. This paper discusses only the programs that explicitly try to model human processes. For a more complete overview of AI models that handle natural language processing, see Tennant (1981).

These models often have a very limited conception of communication, because they are developed by computer scientists and not by human communication researchers. For example, SAM, PAM, and TALE-SPIN (Schank & Riesbeck, 1981), programs that derive from Charniak's (1972) story comprehension model, are all natural language programs that manipulate story texts in one way or another. Telling and listening to stories is certainly a communicative activity (Jefferson, 1978; Ryave, 1978), but hardly the prototypal case of human interpersonal interaction. The exception is a program like PARRY (Faught, Colby, & Parkison, 1974; Faught, 1977) that engages in dialog by modeling the verbal behavior of a paranoid patient as the human partner role-plays a psychiatric interviewer. The conversational mode is a more common context for human communication, but the rest of the model is far from standard. That is not to say that studying abnormal interactional patterns has no value (Labov & Fanshel, 1977), but that requests made during normal cooperative conversation are more typical than therapeutic discourse.

Despite the objections, these programs represent two trends in which AI programs obviously have a formal equivalence to human communication theories, and have a potential dynamic equivalence to human communication processes. These two trends involve systems of allied knowledge structures versus systems of contrasting knowledge structures. Consequently, this

section is divided into two parts. First, AI models and communication-related theories that use allied knowledge structures are examined. Then, AI models and communication-related theories that use contrasting knowledge structures are presented.

Frames

The first trend is represented by such programs as SAM and PAM (Schank & Riesbeck, 1981), and by the programming languages FRL (Roberts & Goldstein, 1977) and KRL (Bobrow & Winograd, 1977). These use cognitive models built on frames (Minsky, 1975; Charniak, 1977), scripts (Schank & Abelson, 1977), or schemas (Simmons & Chester, 1982), alternative terms for the same concept. A script is defined roughly as a knowledge structure that defines an everyday occurrence, usually composed of stereotyped sequences of events. Some typical scripts either developed or proposed have included eating in a restaurant, taking a subway ride, or going to a museum. For example, SAM (Script Applier Mechanism) is an AI system that uses knowledge about the context of everyday occurrences to aid in the understanding of stories. PAM (Plan Applier Mechanism), rather than simply chronicling sequences of events, involves reasoning about people's intentions to find explanations for a character's actions.

A variety of theories related to human communication research uses similar concepts and terminology. Certain communication theories evolve from or rely on cognitive socio-psychological theories, like the analysis of causal schemata by Kelley (1971) or of interpretive schemas by Stotland and Canon (1972). But these AI models do not relate only to cognitive theories of human communication. Erving Goffman, the noted symbolic interactionist, in his book, *Frame Analysis* (1974), argues that

> . . . definitions of a situation are built up in accordance with principles of organization which govern events--at least social ones--and our subjective involvement in them; frame is the word I use to refer to such of these basic elements as I am able to identify. (Goffman, 1974, pp. 10-11)

For example, one cannot point to a "stolen base" as one can point to a dog or cat. To understand the concept of a "stolen base," a group of people must invent or build up a knowledge frame about the elements and rules of the social interaction known as the game of baseball.

Scales

An alternative to such models of allied knowledge structures are models that concentrate on contrasting knowledge structures, exemplified by TALE-SPIN and PARRY. TALE-SPIN is an AI program that writes simple stories about a small world of characters who are motivated to act by having problems to solve. Included in the approaches to solving problems is the choice to use persuasion. Such an alternative requires information about social relationships and personal characteristics, stored as "scales," such as honesty, affection, and energy (the "energy" scale is composed of seven levels: wiped out, exhausted, tired, rested, lively, energetic, and hyper). PARRY is composed of a pattern-matching pre-parser that decomposes the user input into a limited number of acceptable patterns and a system of scales that represent psychological states of the paranoid individual being modeled. Certain patterns cause the values of these scales to change, affecting the normality or abnormality of the model's utterances or actions. PARRY is an outstanding example of how natural and human-appearing an AI model of conversational interaction can be, and TALE-SPIN produces very human-like stories on about the level of some of the simpler of Aesop's fables. Unlike the script or frame approach, however, the scale approach is not extensively developed, certainly not to the point where it has a full programming language designed around the concept.

Examples of human communication theories that use such scale representations also are plentiful. A recent presentation of the constructivist approach to communication theory and research says:

> Our conception of cognitive processes draws heavily on Kelly's (1955) theory of personal constructs. . . . Constructs (e.g., friendly-unfriendly, tall-short, etc.) are the contrasts that persons use to group events. (Delia, O'Keefe, & O'Keefe, 1982, p. 151)

Such a comparing-contrasting representation of cognitive structuring can be seen also in the theory related to the semantic differential (Osgood, Suci, & Tannenbaum, 1957), and the research method using Likert scales (Likert, 1932) or similar scalar systems.

Rule-based AI and Communication Research

This section of the paper examines the area of AI that is currently the fastest growing and probably most important contribution of AI research. Also, this area of AI has the greatest potential for affecting human communication research. This is rule-based programming, often called production systems or systems of production rules. This section is subdivided into three major parts. First, rule-based programming is compared with more traditional procedural programming to see the advantages of rule-based programming for human communication research. Second, we examine the potential equivalency between rule-based programming and the rules perspective in human communication research. Finally, an overview of some current rule-based AI models is presented.

Rule-based Computer Programming

Rule-based programming, often called production systems or systems of production rules, was first proposed by Post (1943). In some ways, rule-based programming differs significantly from traditional procedural approaches embodied in languages like BASIC and FORTRAN. At the same time, rule-based programming maintains computational equivalence with those traditional languages, i.e., it has been demonstrated that anything computable by procedural approaches is also computable by rule-based approaches (Post, 1943; Markov, 1954). Some computer scientists argue that because procedural-based and rule-based languages are computationally equivalent, there is no reason to adopt a rule-based approach. Although a given problem has an infinite number of equivalent specifications, it is well known that certain of those alternative specifications make it easier for humans to solve that problem than other specifications (Haugeland, 1981). For example, the language of mathematics used by the Romans, commonly called Roman numerals, was adequate for recording numbers, but made addition difficult and multiplication practically impossible. The language of mathematics used today, developed by the Arabians, is equal to Roman numerals for dealing with the problem of recording numbers, but far superior when it comes to allowing one to engage in higher-level computations.

Alternative Rule-based Programming Languages: There are several general purpose, rule-based computer programming languages. The oldest is PROLOG (for PROgramming in LOGic); two others are OPS-5 (Forgy & McDermott, 1977) and ROSIE (Fain, Hayes-Roth, Sowizral, & Waterman, 1982). PROLOG was invented in the early 1970s in France, but there were few papers in English describing the language until the late 1970s (one exception is Kowalski, 1974). Despite its relative newness, its impact has been dramatic. Already, the Japanese have selected PROLOG to be the basic model for the internal design of their "Fifth Generation" computers (Warren, 1982). There are several PROLOG implementations available, each with a slightly different syntax, different built-in functions or rules, and different names. Other versions include Q-LOG (Komorowski, 1982), LOGLISP (Robinson & Sibert, 1980), HCPRVR (Chester, 1979), and HC (Roach & Fowler, 1982). The reasons PROLOG is being suggested over OPS-5 and ROSIE as the standard for expressing communication rules include the fact that PROLOG is the oldest general purpose rule-based programming language; the impact PROLOG has had; and, therefore, that PROLOG implementations are more readily available on a variety of computers, including microcomputers like the IBM PC and Apple Macintosh.*

FORTRAN vs. PROLOG: To appreciate the value of rule-based programming to human communication research, it will be helpful to understand some important differences between procedural languages and rule-based languages. This section presents an example of PROLOG code used to parse a subset of English sentences syntactically. Consider a simple, familiar transformational grammar:

*Macintosh is a trademark of Apple Computers, Inc.

```
sentence → noun-phrase, verb-phrase.
noun-phrase → determiner, noun.
noun-phrase → proper-name.
verb-phrase → verb, prepositional-phrase.
prepositional-phrase → preposition, noun-phrase.
determiner → THE.
noun → CAT.
proper-name → BOWIE.
verb → IS.
preposition → ON.
noun → MAT.
```

This grammar allows a simple sentence, like "THE CAT IS ON THE MAT," to be parsed correctly. But how does one create an automatic, computerized parser to do this sort of work?

Before the PROLOG rules are given, a word about the syntax should be said. Most people are familiar with infix notation in which an addition statement, for example, is expressed A + B = C. Most PROLOG code uses prefix notation, however, expressing that statement as +(A B C). Also, since PROLOG is a computer language, it is necessary to distinguish between variables and constants. All variables will be words beginning with a question mark (?) in this syntax. Therefore, the rules for this parser in PROLOG are:

```
sentence(?word-list [?v [actor  ?np]  |  ?pp] ?rest) →
            noun-phrase(?word-list  ?np  ?rest1),
            verb-phrase(?rest1  [?v  |  ?pp]  ?rest),
            number(?np  ?num),
            number(?v  ?num).
noun-phrase([?word1  |  ?rest1] [?n [det  ?word1]] ?rest) →
            category(?word1  determiner),
            noun(?rest1  ?n  ?rest).
noun-phrase([?word1  |  ?rest1] [?word1 [name  ?feat]] ?rest1) →
            category(?word1  proper-name),
            feature(?word1  ?feat).
noun([?n  |  ?rest] ?n  ?rest) →
            category(?n  noun).
verb-phrase(?word-list [?v  ?pp] ?rest) →
            verb(?word-list  ?v  ?rest1),
            prep-phrase(?rest1  ?pp  ?rest).
verb([?word1  |  ?rest] ?word1  ?rest) →
            category(?word1  verb).
prep-phrase([?word1  |  ?rest1] [?feat [?word1  ?np]] ?rest) →
            category(?word1  preposition),
            feature(?word1  ?feat),
            noun-phrase(?rest1  ?np  ?rest).
feature([?word1  |  ?rest] ?feat) →
            feature(?word1  ?feat).
number([?word1  |  ?rest] ?num) →
            number(?word1  ?num).
```

A small dictionary to go with this parser follows:

```
category( A       determiner).
category( ARISES verb).
number(   ARISES singular).
category( BOWIE  proper-name).
```

```
feature(  BOWIE  person).
number(   BOWIE  singular).
category( CAT    noun).
number(   CAT    singular).
category( FROM   preposition).
feature(  FROM   direction).
category( IS     verb).
number(   IS     singular).
category( MAT    noun).
number(   MAT    singular).
category( ON     preposition).
feature(  ON     location).
category( ROCKET noun).
number(   ROCKET singular).
category( SKY    noun).
number(   SKY    singular).
category( THE    determiner).
category( TOWARD preposition).
feature(  TOWARD direction).
```

We begin the explanation by starting at the bottom of the set of rules defining the parser. The last two rules of the parser, the ones named "feature" and "number," are utility rules saying that the number or feature associated with a list of words and structures is the number or feature of the first word in the list.

The other rules, defining a sentence, noun-phrase, noun, verb-phrase, verb, and prepositional-phrase, all have the same structure in the conclusion, the part of the rule before the "→." Each rule is doing one of two things: it is either taking a list of words and removing enough words at the beginning of the list to construct a representation of the structure being defined by the rule, leaving unused words at the end of the list; or it is taking a representation of the structure being defined by the rule and generating a list of words meeting that definition and adding them to the front of a previous list of words. For example, the "sentence" rule takes a list of words, removes enough words off the front of the list to represent the noun-phrase, creates that noun-phrase representation, gives the remaining words to the verb-phrase to be used to create a verb-phrase representation, and is hopefully left with no words remaining. Or it moves in the other direction, generating a sentence from an appropriate representation. Each rule, therefore, has three basic arguments: the first is a list of words, the second is the parse of the beginning of the list of words from the first argument, and the third is a list of words left over after removing the words needed from the first list to create the structural description in the second list. This is necessary since the rules are not merely guidelines for parsing and generating sentences, as is the case with the transformational grammar given above, but programming code to actually do the parsing or generating. These three arguments constitute the input and output variables of the rules.

The definition of the rules come mainly after the "→." The "sentence" rule is defined as a noun-phrase followed by a verb-phrase, with an additional test checking whether the number of the noun-phrase and verb-phrase agree; this test did not exist in the original transformational grammar. The "verb-phrase" rule is defined as a verb followed by a prepositional-phrase. A verb is a word that is categorized in the dictionary as a verb. And so on for the other rules.

This set of rules allows one to test whether a list of words is a sentence, according to this grammar, by entering the list of words in the "sentence" rule as the first argument:

```
sentence([THE CAT IS ON THE MAT] ?parse-answer []).
```

Notice that the third argument is a list with nothing in it. This is because the first argument is a sentence only if it can be parsed with nothing left over. The result of this inquiry is the following:

```
sentence([THE CAT IS ON THE MAT]
         [IS [actor [CAT [det THE]]]
             [location [ON [MAT [det THE]]]]]
         []).
```

On the other hand, if one wants to generate a sentence, one simply enters an appropriate representation in the second argument:

```
sentence(?sentence
         [ARISES [actor [ROCKET [det A]]]
                 [direction [TOWARD [SKY [det THE]]]]]
         []).
```

and receives the response:

```
sentence([A ROCKET ARISES TOWARD THE SKY]
         [ARISES [actor [ROCKET [det A]]]
                 [direction [TOWARD [SKY [det THE]]]]]
         []).
```

With the small dictionary given, many sentences can be parsed and generated, including:

```
[A CAT ARISES FROM THE MAT]
[THE CAT IS ON BOWIE]
[BOWIE IS FROM THE SKY].
```

At this point, it will be useful to say more about the syntax of PROLOG. There are many potential forms for rules, including modal logic, first-, second-, and ω-order predicate calculus. PROLOG statements are a particular type of first-order predicate calculus that contain exactly one conclusion; these statements are termed Horn clauses (Horn, 1951). It can be demonstrated that any statement that can be expressed in first-order logic can be expressed in Horn clauses (Kowalski, 1979, p. 16).

Consider trying to write such a parser in FORTRAN. One would have to program a pointer system, probably using array indices, to keep track of the words; a stack to test sequentially each rule part; a pattern matcher to decide if the current set of words fulfills the definition of the transformational rule; etc. Also consider the problem of extending the parser. The rule defining a verb-phrase allows one to have only a verb followed by a prepositional phrase. But a verb-phrase may be simply a verb, or it may be a verb followed by a direct and/or indirect object. To extend such a FORTRAN system, one would have to re-write large parts of the FORTRAN code. To extend this PROLOG system, one simply adds the extra rules for the verb-phrase.

The code for this parser in FORTRAN would far exceed the size limitations for this paper; the 50 lines of PROLOG given above would require several thousand lines of FORTRAN code to do the same things. In truth, the best way to have FORTRAN do this kind of symbolic manipulation is to write a PROLOG interpreter or compiler in FORTRAN and use the FORTRAN-based PROLOG to do the symbolic manipulations. This certainly has been done; although the original implementation of PROLOG was written in ALGOL-W, the second and most influential version of PROLOG was written in FORTRAN (Battani & Meloni, 1973; Roussel, 1975). Both FORTRAN and PROLOG do "number-crunching" in fairly simple ways, whereas PROLOG's advantages include its combination of syntactical simplicity with high-level symbolic-manipulation power. In this sense, the difference between FORTRAN and PROLOG is analogous to the difference between Roman numerals and Arabic numerals; both FORTRAN and PROLOG handle simple symbolic (mathematical) tasks easily, but PROLOG is far superior when doing higher-level symbolic computations.

Procedural vs. Declarative Languages: There are some obvious differences between the FORTRAN and PROLOG code. First, in FORTRAN one must consciously program the flow of control, i.e., the lines are at least implicitly numbered and the commands are enacted sequentially

unless explicitly redirected by a DO loop or a GO TO or IF command. In rule-based programming, the flow of control is handled entirely by the PROLOG interpreter or compiler, so there are no explicit DO loops or GO TO commands. This does introduce a problem for some people. FORTRAN solves many types of problems by using iteration, i.e., the value of a counter variable is changed, and the procedure is repeated using the new value of that variable until some cut-off value is reached. Since PROLOG has no flow of control explicitly stated by the programmer, iteration is rarely used; PROLOG uses recursion to do the same sorts of tasks, i.e., a rule may "call" on itself to create a definition. For people familiar with iteration and unfamiliar with recursion, the change in programming style may prove troublesome; but the rewards are immense.

A second difference is that FORTRAN is entirely procedural, i.e., to understand the code, one must be familiar with the syntax of FORTRAN and the flow of iterative statements. PROLOG is declarative in nature. That is, with a little practice with the syntax one can easily read the rules as a definition of the concept. The rules are, however, not only declarative definitions but also programs that can be executed by a computer with the appropriate interpreter or compiler. Like FORTRAN, therefore, PROLOG code is instruction to a computer to do certain types of processing. But a theory or model expressed as a system of related Horn clauses is unique in being a static representation that can be unambiguously and consistently interpreted by different people (unlike many vague, verbal theories in social science). Recall that the 50-line parser given above would require thousands of lines of FORTRAN code to do the same sorts of things, code which would be difficult for a person to read and understand. And when a computer interprets the system of logical statements, it is an active process that is a working model of the process under study (a further discussion of working models of communication theories is in Sanford and Roach, 1984). More is said about the advantages of having communication rules in a form interpretable by computers in the section titled "Example Rule-based AI Models."

Rules Perspective on Human Communication

Equivalence between Communication Rules and Rule-based Programming: The primary argument here is first, that rule-based computer languages have a formal equivalence to communication rules (Cushman & Whiting, 1972; Pearce, 1976); and second, that rule-based computer languages are automatic formal systems with the potential of having a dynamic equivalence to the rule-governed behavior of communicating human beings (Haugeland, 1981).

Now, in what sense are these PROLOG statements rules? Most of us are used to thinking of rules as "if . . . then" statements. For example, Pearce, Cronen, Johnson, Jones, & Raymond (1980) and Cronen, Pearce, & Harris (1982) present primitive forms of rules, one of which is reproduced with slight modification in Figure 1. For PROLOG, that order is reversed; the "if" part of the rule is listed after the "then" part. As mentioned above, the advantage to such a rule-based language is not entirely obvious when one is "number-crunching." But one may clearly see the syntactic clarity of PROLOG when one translates the example in Figure 1 as follows:

```
counts-as(insult  gamesmanship)  if
                                 episode(playing-the-dozens),
                                 antecedent(opponent's-turn).
```

This can be read as: {an insult counts as gamesmanship if (the episode is playing the dozens) and (the antecedent is the opponent's turn)}. This rule is not only a representation of the rules-theoretic concept in human communication research, but is also programming code with which a computer can logically manipulate the symbolic values of the concept. That is, not only do rules in this form constitute a static, theoretical statement that can be unambiguously interpreted by a human researcher, but also they embody a formal, logical specification that can be interpreted by a computer compiler or interpreter to produce a working model in action.

Notice that the PROLOG code given in the example above has no variables, that is, none of the literals begins with a question mark. This means that the statement expresses a fact; if the above rule is asserted, one is asserting a simple fact. If the above rule is presented as an inquiry and the clauses after the "if" are true, the answer returned by the PROLOG interpreter or compiler about the "then" part will be true; if any of the clauses are false, the conclusion will be false.

Figure 1

Primitive Form of a Constitutive Rule (Adapted from Cronen et al., 1982, p. 74)

$$cR = \overline{\left| \; A \; > \; [MC^1 \rightarrow MC^2] \; \right|}^{MC^3}$$

Where:

cR	=	Constitutive Rule
A	=	Antecedent Condition
MC	=	Meaningful Construction
^{1,2,3}	=	Levels of Abstraction
$\rightarrow$	=	Read "counts as"
>	=	Read "if . . . then"
$\overline{}$	=	Read "in the context of"

Example:

Episode: Playing the Dozens

$$cR^1 = \overline{\left| \; \text{Opponent's Turn} \; > \; [\text{Insult} \rightarrow \text{Gamesmanship}] \; \right|}$$

Types of Logic: There is an important difference between the type of logic used by communication rules theorists and rule-based computer programming. Some scholars have suggested using the standard form of first-order predicate logic (Schenck-Hamlin, Georgacarakos, & Wiseman, 1982), but the notation system is so esoteric that it has not been widely adopted. Generally, communication scholars know and use a form of modal logic. For example, most communication rules theorists recommend deontic logic or the practical syllogism (Cushman & Pearce, 1977; Shimanoff, 1980), in which certain important logical relationships (i.e., action A is "required" if condition B, action A is "prohibited" if condition B, action A is "allowed" if condition B, action A is "intended" if condition B) are most naturally and directly expressed. These are two among many modal logics, although these are non-standard modal logics since they are systems without the axiom of logical necessity (Hughes & Cresswell, 1968). They make it very easy to express these four relationships, but there is no proof theory associated with any modal logic and therefore no way to form automatic deductions from such statements. According to Moore (1980), however, the axioms of modal logic can be expressed in Horn clauses. There is a proof theory associated with Horn clauses, so automatic deduction techniques can be applied, as the programming language PROLOG demonstrates. This translation of communication rules from modal (propositional) logic to Horn clause (predicate) logic would give researchers the advantage of being able to use this proof theory to deduct logically valid statements from the axioms of a theory specified in predicate logic.

An example of the difference between propositional and predicate logics might help explain the discussion above. When one takes logic as a course in philosophy, usually one is taught propositional logic. That is, the variables of the logical expressions are entire propositions or statements. Then one is taught to construct truth tables: if statement P is true and statement Q is true, then statement (P and Q) is true and statement (P or Q) is true. Say we have applied the

above principle to an earlier set of true statements so that we have the following statement as P and it is true:

```
P:  (Adam is the father of Bill) and (Bill is the father of Charlie)
```

Now we also assert that the following statement Q is true:

```
Q:  (a person is the grandfather of another person if the first
    person is the father of a third person and the third person
    is the father of the second person)
```

All we may conclude from the above is that the statement (P and Q) is also true. The statement (P and Q) is the following:

```
(P and Q):  ((Adam is the father of Bill) and (Bill is the father of
            Charlie)) and (a person is the grandfather of another
            person if the first person is the father of a third person
            and the third person is the father of the second person)
```

This gets one nowhere. What one would like to conclude is that Adam is the grandfather of Charlie.

This can be deduced automatically if the propositions are translated into predicate logic. In PROLOG, the logical statements above are the following:

```
father(Adam  Bill).
father(Bill  Charlie).
grandfather(?person1  ?person2) if
                                father(?person1  ?person3),
                                father(?person3  ?person2).
```

Notice that entire statements or propositions are not the variables, but that individual terms are variables. With this set of logical statements, one achieves the desired answer. If one asks who is the grandfather of whom:

```
grandfather(?person1  ?person2).
```

one receives the answer:

```
grandfather(Adam  Charlie).
```

Or one may ask the simple question whether Adam is the grandfather of Charlie:

```
grandfather(Adam  Charlie).
```

and receive the answer that the statement is true. The deduction of this "new" fact, beyond the statements originally asserted, is possible only by using a form of logic that has a proof theory associated with it, possible only by using predicate logic.

Also note the problem with the most familiar logical structure in communication research, the syllogism, adapted by communication rules researchers into the logically weaker form called the practical syllogism. Try to translate the logical statements about kinship given above into syllogistic form so that you can arrive at the conclusion that Adam is the grandfather of Charlie. The task is almost impossible, since syllogisms are not as general as Horn clauses. Indeed, any statement that can be expressed in first-order logic can be expressed in Horn clauses (Kowalski, 1979; Moore, 1980).

Example Rule-based AI Models

Because PROLOG is such a recent development in rule-based programming, very few models of human communication processes written in PROLOG are available. Some of the few exceptions include the works of Dahl and Saint-Dizier (1985), Colmerauer (1982), Simmons and Chester (1982), and Warren and Pereira (1982). However, other rule-based examples abound (Waterman & Hayes-Roth, 1978; Buchanan & Duda, 1982). In these non-PROLOG examples, the form of the rules was created individually by each researcher to fit the particular problem being addressed. This means that these production-rule systems are not written as general problem-solvers but are very restrictive in application, a problem that helps contribute to their inability to model the full complexity of human communication processes.

Despite these shortcomings, we believe that AI models have had remarkable success at modeling human communication processes. Programs such as PARRY (Faught, 1977; Faught, Colby, & Parkison, 1974), a program that models the verbal behavior of a paranoid patient while the human user role-plays a psychiatric interviewer, have a naturalness and facility that belies their artificiality. And when other rule-based production systems are included, the potential power is better appreciated. For example, MYCIN (Davis, Buchanan, & Shortliffe, 1977; Shortliffe, 1976) is not capable of engaging in impressive conversational dialogues, but is a medical diagnostician that rivals the best human doctors. Other expert-systems that impressively model human communication processes using rule-based programming include EXPERT (Weiss & Kulikowski, 1979), RAINBOW (Hollander & Reinstein, 1979), and HEARSAY (Erman, London, & Fickas, 1981), among others.

The current work we are pursuing involves modeling the communication rules of interpersonal request forms, discussed here in the context of interaction between a sales agent and a customer, focusing on the rules of the agent. The methodology of this research is more fully elaborated in Sanford and Roach (1984). The rules below display only a part of the full rule system that constitutes the model of communicating requests. It begins with the top-level rule for how an agent interprets and responds to an utterance from a customer. Before examining the clauses of the rule below, one should first understand the abbreviations of rule and data names used here. This rule system presents the rules for an agent, i.e., the rules are written from the perspective of the salesperson. The agent, therefore, is the first person (person 1); the customer is the second person (person 2); a third person perspective does not exist at this level. The communication is explicitly divided into content and relationship components (Watzlawick, Bavelas, & Jackson, 1967). This means that "content2" stands for the content of the customer's message (the content of person 2), "relation2" stands for the interpersonal relationship expressed by the customer (the relationship expressed by person 2), "content1" is the content of the agent's message, and "relation1" is the interpersonal relationship expressed by the agent.

The process of responding to another person's utterance may be broken down into three stages. The first stage is to decode the other's statement, to understand what the other has said. The second stage is to plan one's own actions, in light of what the other just said. And the third stage is to actually do (i.e., say, write, or enact) what one has planned.

```
agent(?input2)  if                                  ;3 stages of
                personality1(?p-relation1),         ;analysis:
                parse2(?input2                      ;---1---
                       ?p-relation1                 ;interpret
                       ?content2                    ;customer's
                       ?relation2),                 ;utterance
                personality2(?relation2             ;& him/her
                             ?p-relation2),         ;---2---
                i-relation1(?p-relation1            ;plan
                            ?p-relation2            ;conversation
                            ?i-relation1),          ;path
                plan-utterance1(?content2           ;&
                                ?i-relation1        ;request
                                ?content1           ;form
                                ?structure1),       ;---3---
                output(?content1                    ;enact
                       ?structure1).                ;plan
```

The rule governing the agent's handling of and response to a customer's utterance is the "agent" rule. As the rule is entered, it brings in the customer's utterance through the variable "?input2." The customer's utterance was read in by a higher-level rule that contains the "agent" rule in the same way this rule contains lower-level rules, like the ones discussed below. The "agent" rule first finds out the agent's general approach to the relationship dimension of communication, by invoking the "personality1" rule. The personality component of the relationship level of communication is returned in the "?p-relation1" variable. The second step interprets or decodes the customer's utterance by invoking the "parse" rule. The input variables to the "parse" rule are "?input2" and "?p-relation1", the first being brought in when entering the "agent" rule and the second coming from the "personality1" rule, while the "?content2" and "?relation2" variables are output by the "parse" rule. A simple "parser" is presented above, in the section titled "FORTRAN vs. PROLOG," as the example definition of a transformational grammar. The third step models the personality of the customer, based on the relationship component of the current utterance in light of the model of the customer from previous interaction. The input variable of the "personality2" rule is "?relation2" and the resulting model of the customer returns through the output variable "?p-relation2." The fourth step determines the intermediate relationship component for this utterance with the "i-relation1" rule. This rule will be discussed more extensively below. The fifth step plans the agent's utterance in response to what the customer said. That is, the fifth step decides two things: where the agent wants to go in the conversation, i.e., what conversational path should be followed; and how to phrase the content of the agent's current utterance. This is done by the "plan-utterance1" rule, that reads in the customer's content in variable "?content2" and the agent's relationship approach in "?i-relation1," then reads out the agent's content in variable "?content1" and the particular request form in "?structure1." If "?i-relation1" is "hard," implying an aggressive or demanding request wording, then possible forms would include need statements and imperatives. If "?i-relation1" is "soft," implying a passive or pleading request wording, then possible forms would include imbedded imperatives and requests-to-request. These are selected from the types of requests identified by Sanford & Roach (1986). And finally, the "agent" rule actually responds to the customer's utterance with the "output1" rule that displays information on the customer's terminal, talks to the customer through a voice simulator, interacts with the customer by controlling a robot, or whatever.

Notice the potential sources of information for determining the agent's relationship component of the current output utterance. The first is the personality of the agent ("?p-relation1"), the second is the personality of the customer based on the customer's behavior ("?p-relation2"), and the third is the context of the conversation (through the "plan-utterance1" rule). To get to the output, the agent's relationship component goes through three stages: the most general, stable level of relationship determination, "?p-relation1"; an intermediate level of relationship analysis, "?i-relation1"; and a final, specific relationship approach, "?structure1."

As an example, let us say that the personality of the agent can be one of four possible types: hard, soft, complementary, or symmetrical (Watzlawick, Bavelas, & Jackson, 1967). The following version of the "i-relation1" rule would suffice for a model of a non-complex agent:

```
i-relation1(hard  ?a  hard).
i-relation1(soft  ?a  soft).
i-relation1(symmetrical  hard  hard).
i-relation1(symmetrical  soft  soft).
i-relation1(complementary  hard  soft).
i-relation1(complementary  soft  hard).
```

As one can see, if the personality of the agent is "hard" or "soft," it does not matter what the personality of the customer is; the intermediate decision is to be "hard" or "soft," respectively. But if the personality of the agent is either "complementary" or "symmetrical," then the choice of the intermediate relationship component of the agent is determined by the personality of the customer.

This "i-relation1" rule provides a model of a very dogmatic individual who sees people in black or white terms. The "personality2" rule could be set for an equally short-sighted individual, like the following:

```
personality2(?relation  ?relation).
```

This combination of "i-relation1" and "personality2" rules would make the agent pay attention only to the latest customer utterance in planning how to respond. This combination represents the simplest of Markov chains; the current form would be based only on the immediately preceding behavior of the other person.

On the other hand, this is not the only way the general model can be specified. Without changing the "agent" rule in any way, the model could be made to represent a more complex individual. Define the "personality2" rule as the following:

```
personality2(?rel  ?p-rel)  if
                            p-model2(?p-model2),
                            match(?p-rel
                                  [?rel  |  ?p-model2]),
                            assert(p-model2(?p-rel)).
```

Note that the "assert" rule exemplified above is a rule that automatically creates a new rule in the system. This makes the "?p-relation2" variable in the "agent" rule return a list of the relationship judgments for all the customer's utterances to this point, with the latest utterance at the front of the list. For example, after the fifth utterance, "?p-relation2" might look like the following:

```
[hard  hard  soft  hard  soft]
```

Then the "i-relation1" rule could be re-written to consider the pattern of customer responses, rather than merely the latest response. For example, the "i-relation1" rule could account for the last two utterances of the customer by being written in the following way:

```
i-relation1(hard  ?a  hard).
i-relation1(soft  ?a  soft).
i-relation1(symmetrical [hard  hard  |  ?a] hard).
i-relation1(symmetrical [hard  soft  |  ?a] soft).
i-relation1(symmetrical [soft  hard  |  ?a] hard).
i-relation1(symmetrical [soft  soft  |  ?a] soft).
i-relation1(symmetrical [soft  |  ?a] soft).
i-relation1(symmetrical [hard  |  ?a] hard).
i-relation1(complementary [hard  hard  |  ?a] soft).
```

```
i-relation1(complementary [hard  soft  |  ?a] hard).
i-relation1(complementary [soft  hard  |  ?a] soft).
i-relation1(complementary [soft  soft  |  ?a] hard).
i-relation1(complementary [soft  |  ?a] hard).
i-relation1(complementary [hard  |  ?a] soft).
```

This extension is still a model of a very non-complex individual. A personality is determined by a limited number of one-word descriptions. For example, an agent's intermediate interpersonal relationship component ("?i-relation2") can be only "hard" or "soft," but there is nothing between these extremes. If a theorist wishes to represent the relationship component as a dimension of intermediate values, as in a Likert scale (Likert, 1932), rather than an either/or choice, the theorist can represent the personality as a sequence of verbal descriptors or as a numeric scale (e.g., "extremely pushy" equals 1, "somewhat pushy" equals 2, "somewhat easy going" equals 3, and "very easy going" equals 4) and re-write the "personality2" and "i-relation2" rules to manipulate the numbers appropriately. And finally, if a theorist wants to model a cognitively complex individual who sees the interpersonal relationship component as several dimensions, the rules can be written to pass more complex, multi-dimensional structures, such as lists of dimension names and values, arrays, or other rules. That is, the rules can be used to represent any combination of quantitative and qualitative dimensions of personality. Since this is an implementation of first-order predicate calculus, any precise, logical theory can be specified.

As these examples show, the rule-based programming approach has the advantage of modularity. Rules can be written, rewritten, added, and subtracted with a minimum disruption of the rest of the rule-base. Clearly, the example rules given above do not do much by themselves, but when embedded in a knowledge-base of many rules, each created one at a time or in small groups or modules, one's ability to form a system with a dynamic equivalence to human communication is greatly enhanced. This should be particularly important to those interested in the rules-perspective on human communication research. We have heard critics of the rules-perspective argue that once a rule has been identified, what does one do with it? Cronen et al. (1982) attempt to answer this criticism when they discuss how a system of rules links to form a logic of conversation (pp. 81-82). If the data-base of related rules is of any significant size, following the linkages and the flow of logic by hand is impractical. But if the theory is presented as Horn clauses, the computer can follow the logic with a rapidity and accuracy impossible to humans. Indeed, some current PROLOG implementations can make 30,000 logical inferences per second (lips), whereas the Japanese Fifth Generation computers are expected to be able to make 5 million lips.

This discussion of the syntax of and programming with PROLOG has been necessarily brief. It is not the purpose of this paper to teach one how to program in PROLOG. For an effective tutorial in PROLOG programming, see Clark and McCabe (1984).

In the previous section on traditional AI techniques, we identified 2 major knowledge structures: the frame structure and the scale structure. Each of those is represented by various theories of human communication. Here, we have identified a third type of knowledge structure, the logical (or rule) structure, represented by the rules-theoretic approach to human communication. Many computer science traditionalists view rule-based programming in just this way, as simply another technique on a level equal with and comparable to the other techniques developed by AI theorists. To us, however, this appears too restrictive a view of production rules. Because these rules are a form of first-order predicate logic, any statement that can be made precise enough can be expressed as Horn clauses; because theories are supposed to be statements about the relationships of a process under study, any precise theory is expressible as a system of Horn clauses, including the theories of cognitivists and interactionists. No PROLOG program using scalar knowledge structures has been developed yet, although that would be quite easy to do; but frames already have been represented in Horn clauses (Simmons and Chester, 1982). Production rules, therefore, have proven to be more than just another AI programming technique; they have led to qualitatively more powerful expert systems, both by using the previously developed programming techniques in more powerful ways and by introducing new techniques at the same time.

Conclusion

This paper was divided into two major sections. The first section examined the power of AI programs to represent 2 types of human communication theories, those based on allied knowledge structures (i.e., schematas, schemas, scripts, or frames) and those based on contrasting knowledge structures (i.e., constructs, semantic differentials, or Likert scales). These apply to symbolic interactionist, sociologic theories of communication as readily as to cognitive theories of communication. The second section examined the newest development in AI programming, rule-based programming, and how PROLOG rules may represent communication rules theory and, indeed, any precisely stated theory, since PROLOG is a representation of first-order predicate calculus.

Some argue that the rules approach to communication theory differs from the systems perspective (Cushman & Pearce, 1977). Others say that they really have much in common (Boynton, 1977; Cronen & Davis, 1978). Implicit in the argument of this paper is an agreement with the latter approach. Systems research allows a variety of foci, depending on the interest of the researcher. An individual, a small group, an organization, a society may all be selected as appropriate systems for study; a system may be closed or open, simple or complex. What is supposed to unify the systems approach is a dedication to mathematical description of the systems of study (Watzlawick, Bavelas, & Jackson, 1967). Logic is the fundamental branch of mathematics, the basis of all other types of mathematics. Horn clauses are logical statements or rules that can be used to unify the diversity of human rules research.

The purpose of this paper is to argue that it would be advantageous to learn and adopt PROLOG as a standard for representing communication rules. At this point, we would like to summarize those advantages. The first is that adopting a standard will make it easier for scholars to understand and communicate with one another about their rules research. The second advantage is that, unlike other types of logic, Horn clauses have a proof theory associated with them, so that theories stated in Horn clauses would yield automatically derivable logical implications. The third advantage is that, unlike current theories that are expressed by vague verbal descriptions and/or imprecise statistical models, theories expressed in PROLOG would be presented in a precise, unambiguous manner that could be consistently interpreted by any scholar. (One can not know how the variables in a regression equation or a Markov analysis relate to one another causally or structurally by examining the statistical model. Since any precise theory can be stated in Horn clauses (Kowalski, 1979; Moore, 1980), any statistical model can be expressed in logic. But there is no reason to limit oneself when logic can capture a much richer structural description than is available from a statistical model.) And the fourth advantage (discussed in more detail in Sanford & Roach, 1984) is that besides being open to unambiguous interpretation by human scholars, a theory expressed in Horn clauses can also be interpreted by a computer. This provides the advantage of having more than a static description of one's theory, but of being able to see the theory transformed into an active process, a working model of the theory in action.

References

Battani, G. & Meloni, H. (1973). *Interpreteur du language de programmation PROLOG.* Marseille, France: Groupe d'Intelligence Artificielle, Université d'Aix-Marseille.

Bobrow, D. G. & Winograd, T. (1977). An overview of KRL, a knowledge representation language. *Cognitive Science, 1*, 3-46.

Boynton, K. R. (1977). Episodic deviation: A precursor to dynamic homeostasis in interpersonal relationships. Unpublished paper presented to the Speech Communication Association Convention in Washington, D.C.

Buchanan, B. G. & Duda, R. O. (1982). Principles of rule-based expert systems (Tech. Rep. STAN-CS-82-926). Stanford, CA: Stanford U., Computer Science Dept.

Charniak, E. (1972). *Toward a model of children's story comprehension* (Tech. Rep. AI TR-266). Cambridge, MA: M.I.T. AI Laboratory.

Charniak, E. (1977). A framed PAINTING: The representation of a common sense knowledge fragment. *Cognitive Science, 1*, 355-394.

Chester, D. L. (1979). Using HCPRVR (Tech. Rep.). Austin, TX: U. of Texas, Dept. of Computer Science.

Clark, K. L. & McCabe, F. G. (1984). *Micro-PROLOG: Programming in logic.* Englewood Cliffs, N.J.: Prentice Hall.

Colmerauer, A. (1982). An interesting subset of natural language. In K. L. Clark & S.-A. Tarnlund (Eds.), *Logic programming*. New York: Academic Press.

Cronen, V. E. & Davis, L. K. (1978). Alternative approaches for the communication theorist: Problems in the laws-rules-systems trichotomy. *Human Communication Research, 4*, 120-128.

Cronen, V. E., Pearce, W. B., & Harris, L. M. (1982). The coordinated management of meaning: A theory of communication. In F. E. X. Dance (Ed.), *Human communication theory*. New York: Harper & Row, 61-89.

Cushman, D. P. & Pearce, W. B. (1977). Generality and necessity in 3 types of theory about human communication, with special attention to rules theory. *Human Communication Research, 3*, 344-353.

Cushman, D. P. & Whiting, G. C. (1972). An approach to communication theory: Toward consensus on rules. *Journal of Communication, 22*, 217-238.

Dahl, V. & Saint-Dizier (Eds.). (1985). *Natural language understanding and logic programming*. New York: Elsevier Science Publishers.

Davis, R., Buchanan, B. G., & Shortliffe, E. H. (1977). Production rules as a representation for a knowledge-based consultation program. *Artificial Intelligence, 8*, 15-45.

Delia, J. G. (1977). Alternative perspectives for the study of human communication: Critique and response. *Communication Quarterly, 25*, 46-62.

Delia, J. G., O'Keefe, B. J., & O'Keefe, D. J. (1982). The constructivist approach to communication. In F. E. X. Dance (Ed.), *Human communication theory*. New York: Harper & Row, 147-191.

Donohue, W. A., Cushman, D. P., & Nofsinger, R. E., Jr. (1980). Creating and confronting social order: A comparison of rules perspectives. *Western Journal of Speech Communication, 44*, 5-19.

Erman, L. D., London, P. E., & Fickas, S. F. (1981). The design and an example use of HEARSAY-III. *Proceedings of the 7th international joint conference on AI*. Menlo Park, CA: American Association for AI, 409-415.

Fain, J., Hayes-Roth, F., Sowizral, H., & Waterman, D. (1982). Programming in ROSIE: An introduction by means of examples (Rand Note N-1646-ARPA). Santa Monica, CA: Rand Corp.

Fairhurst, G. T. (1980). Using the functional prerequisites to communication rules as a structure for rule-behavior research. Paper presented to the International Communication Association Convention in Acapulco, Mexico. Available from ERIC: Order Document No. ED 194 932.

Faught, W. S. (1977). Motivation and intensionality in a computer simulation model (AI Memo 305). Stanford, CA: Stanford U., Computer Science Dept.

Faught, W. S., Colby, K. M., & Parkison, R. C. (1974). The interaction of inferences, affects, and intentions in a model of paranoia (AI Memo 253). Stanford, CA: Stanford U., Computer Science Dept.

Forgy, C. & McDermott, J. (1977). OPS, a domain-independent production system language. *Proceedings of the 5th international joint conference on AI.* Pittsburgh, PA: Carnegie-Mellon U., Dept. of Computer Science, 933-939.

Frentz, T. S. & Farrell, T. B. (1976). Language-action: A paradigm for communication. *Quarterly Journal of Speech, 62,* 333-349.

Ganz, J. S. (1971). *Rules: A systematic study.* Paris: Mouton.

Goffman, E. (1974). *Frame analysis: An essay on the organization of experience.* New York: Harper & Row.

Haugeland, J. (1981). Semantic engines: An introduction to mind design. In J. Haugeland (Ed.), *Mind design.* Cambridge, MA: M.I.T. Press, 1-34.

Hollander, C. R. & Reinstein, H. C. (1979). A knowledge-based application definition system. *Proceedings of the 6th international joint conference on AI.* Stanford, CA: Stanford U., Computer Science Dept., 397-399.

Horn, A. (1951). On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic, 16,* 14-21.

Hughes, G. E. & Cresswell, M. J. (1968). *An introduction to modal logic.* London: Methuen & Co.

Jefferson, G. (1978). Sequential aspects of storytelling in conversation. In J. Schenkein (Ed.), *Studies in the organization of conversational interaction.* New York: Academic Press.

Kelley, H. H. (1971). Causal schemata and the attribution process. In E. E. Jones, D. E. Kanouse, H. H. Kelley, R. E. Nisbett, S. Valins, & B. Weiner, (Eds.), *Attribution: Perceiving the causes of behavior.* Morristown, NJ: General Learning Press.

Kelly, G. A. (1955). *The psychology of personal constructs* (2 vols.). New York: Norton.

Komorowski, H. J. (1982). QLOG--The programming environment for PROLOG in LISP. In K. L. Clark & S.-A. Tarnlund (Eds.), *Logic programming.* New York: Academic Press, 315-322.

Kowalski, R. A. (1974). Predicate logic as programming language. *Proceedings of the international federation for information processing.* Amsterdam: North Holland Publishing Co., 569-574.

Kowalski, R. A. (1979). *Logic for problem solving.* New York: North Holland Publishing Co.

Labov, W. & Fanshel, D. (1977). *Therapeutic discourse: Psychotherapy as conversation.* New York: Academic Press.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology, 140,* 1-55.

Markov, A. A. (1954). [*The theory of algorithms*] (J. J. Schorr-kon, trans.) (No. OTS 60-51085) Washington, D.C.: U.S. Dept. of Commerce, Office of Technical Services, 1961. (Originally published, Moscow: Academy of Sciences of the U.S.S.R., Mathematical Institute, *42*).

Minsky, M. (1975). A framework for representing knowledge. In P. Winston (Ed.), *The psychology of computer vision.* Hightstown, N.J.: McGraw-Hill.

Moore, R. C. (1980). *Reasoning about knowledge and action* (Tech. Note 191). Menlo Park, CA: SRI International.

Murdock, J. I., Bradac, J. J., & Bowers, J. W. (1984). Effects of power on the perception of explicit and implicit threats, promises, and thromises: A rule-governed perspective. *Western Journal of Speech Communication, 48,* 344-361.

Nofsinger, R. E., Jr. (1975). The demand ticket: A conversational device for getting the floor. *Speech Monographs, 42,* 1-9.

Osgood, C. E., Suci, G. J., & Tannenbaum, P. H. (1957). *The measurement of meaning.* Urbana, IL: U. of Illinois Press.

Pearce, W. B. (1976). The coordinated management of meaning: A rules-based theory of interpersonal communication. In G. R. Miller (Ed.), *Explorations in interpersonal communication.* Beverly Hills, CA: Sage Press, 17-35.

Pearce, W. B., Cronen, V. E., Johnson, K., Jones, G., & Raymond, R. (1980). The structure of communication rules and the form of conversation: An experimental simulation. *Western Journal of Speech Communication, 44,* 20-34.

Post, E. L. (1943). Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics, 65,* 197-215.

Riesbeck, C. K. (1978). An expectation-driven production system for natural language understanding. In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems.* New York: Academic Press.

Roach, J. W. & Fowler, G. S. (1982). The HC manual: Virginia Tech PROLOG (Tech. Rep.). Blacksburg, VA: Virginia Tech, Dept. of Computer Science.

Robert, H. M. (1970). *Robert's rules of order newly revised.* Enlarged by S. C. Robert (Ed.). Glenview, IL: Scott, Foresman & Co.

Roberts, R. B. & Goldstein, I. P. (1977). The FRL primer (Memo 408). Cambridge, MA: M.I.T. AI Laboratory.

Robinson, J. A. & Sibert, E. E. (1980). LOGLISP--An alternative to PROLOG (Tech. Rep.). Syracuse, NY: Syracuse U., School of Information and Computer Science.

Roussel, P. (1975). *PROLOG: Manuel de reference et d'utilization.* Marseille, France: Groupe d'Intelligence Artificielle, Université d'Aix-Marseille.

Ryave, A. L. (1978). On the achievement of a series of stories. In J. Schenkein (Ed.), *Studies in the organization of conversational interaction.* New York: Academic Press.

Sanford, D. L. & Roach, J. W. (1984). The application of artificial intelligence as a methodology for human communication research. Unpublished paper presented to the Southern Speech Communication Association Convention in Baton Rouge, LA.

Sanford, D. L. & Roach, J. W. (1986). Representing and using metacommunication to control speakers' relationships in natural language dialogue. *Proceedings of the third annual USC computer science symposium on knowledge based systems: Theory and applications.* Columbia, SC: University of South Carolina, Dept. of Computer Science.

Schank, R. C. & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding.* Hillsdale, N.J.: L. Erlbaum Associates.

Schank, R. C. & Riesbeck, C. K. (Eds.) (1981). *Inside computer understanding: Five programs plus miniatures.* Hillsdale, N.J.: L. Erlbaum Associates.

Schenck-Hamlin, W. J., Georgacarakos, G. N., & Wiseman, R. L. (1982). A formal account of interpersonal compliance-gaining. *Communication Quarterly, 30,* 173-180.

Shimanoff, S. B. (1980). *Communication rules: Theory and research.* Beverly Hills, CA: Sage Publications.

Shortliffe, E. H. (1976). *Computer-based medical consultations: MYCIN.* New York: Elsevier Science Publishers.

Simmons, R. F. & Chester, D. L. (1982). Relating sentences and semantic networks with procedural logic. *Communications of the Association for Computing Machinery, 25,* 527-547.

Stotland, E. & Canon, L. K. (1972). *Social psychology: A cognitive approach.* Philadelphia: W. B. Saunders.

Tennant, H. R. (1981). *Natural language processing.* New York: Petrocelli.

Warren, D. H. D. (1982). A view of the fifth generation and its impact. *AI Magazine, 3*(4), 34-39.

Warren, D. H. D. & Pereira, C. N. (1982). An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics, 8,* 110-119.

Waterman, D. A. & Hayes-Roth, F. (Eds.) (1978). *Pattern-directed inference systems.* New York: Academic Press.

Watzlawick, P., Bavelas, J. H., & Jackson, D. D. (1967). *Pragmatics of human communication: A study of interactional patterns, pathologies, and paradoxes.* New York: W. W. Norton.

Weiss, S. M. & Kulikowski, C. A. (1979). EXPERT: A system for developing consultation models. *Proceedings of the 6th international joint conference on AI.* Stanford, CA: Stanford U., Computer Science Dept., 942-947.